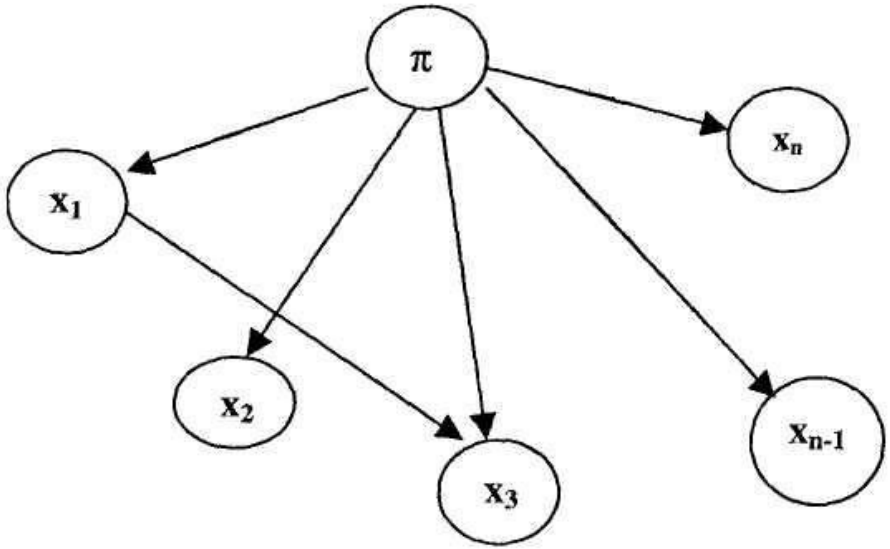
Лабораторная работа: Bayesian Networks на примере датасета Mushroom Classification

Выполнила Жиденко Виктория Александровна
Группа М8О-307Б-23

Bayesian Networks

Байесовские сети — это графовые модели, которые показывают, как переменные связаны между собой через вероятности. Они помогают моделировать зависимости и делать выводы на основе данных.
Пример: предсказать, съедобен ли гриб, зная его запах и цвет жабр.
Формула: $P(A|B)=P(B|A) \cdot P(A)/P(B)$ — закон Байеса для расчёта вероятностей.



Датасет Mushroom Classification

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color
0	p	x	s	n	t	p	f	c	n	k
1	e	x	s	y	t	a	f	c	b	k
2	e	b	s	w	t	l	f	c	b	n

stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
s	w	w	p	w	o	p	k	s	u
s	w	w	p	w	o	p	n	n	g
s	w	w	p	w	o	p	n	n	m

- Источник: Kaggle
- Размер: 8124 гриба
- Признаки: форма шляпки, запах, цвет жабр, место роста и др. (всего 22)
- Цель: классификация на съедобные (52%) и ядовитые (48%)
- Пример: запах "odor" часто указывает на яд.

Загрузка датасета и основная статистика

```
import pandas as pd

path = 'mushrooms.csv'
data = pd.read_csv(path)
```

```
data.shape
```

```
(8124, 23)
```

- 8124 записей, 23 столбца (размер датасета)
- Все признаки — строки (object)
- Целевой столбец: class → e (съедобный), p (ядовитый)

```
data.head(3)
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color
0	p	x	s	n	t	p	f	c	n	k
1	e	x	s	y	t	a	f	c	b	k
2	e	b	s	w	t	l	f	c	b	n

stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
s	w	w	p	w	o	p	k	s	u
s	w	w	p	w	o	p	n	n	g
s	w	w	p	w	o	p	n	n	m

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   class                     8124 non-null   object
 1   cap-shape                 8124 non-null   object
 2   cap-surface               8124 non-null   object
 3   cap-color                 8124 non-null   object
 4   bruises                   8124 non-null   object
 5   odor                      8124 non-null   object
 6   gill-attachment           8124 non-null   object
 7   gill-spacing              8124 non-null   object
 8   gill-size                 8124 non-null   object
 9   gill-color                8124 non-null   object
 10  stalk-shape               8124 non-null   object
 11  stalk-root                8124 non-null   object
 12  stalk-surface-above-ring  8124 non-null   object
 13  stalk-surface-below-ring  8124 non-null   object
 14  stalk-color-above-ring    8124 non-null   object
 15  stalk-color-below-ring    8124 non-null   object
 16  veil-type                 8124 non-null   object
 17  veil-color                8124 non-null   object
 18  ring-number               8124 non-null   object
 19  ring-type                 8124 non-null   object
...
 21  population                8124 non-null   object
 22  habitat                   8124 non-null   object
dtypes: object(23)
```

Обработка датасета

1. Сначала удалили два «плохих» столбца:

- veil-type — везде одинаковое значение 'p', ничего не даёт
- stalk-root — 2480 пропусков в виде знака «?», слишком много

```
data['veil-type'].value_counts()
```

✓ 0.0s

```
veil-type
p    8124
Name: count, dtype: int64
```

```
data = data.drop(['stalk-root', 'veil-type'], axis=1)
```

```
data.shape
```

```
(8124, 21)
```

```
Пропуски по столбцам:
stalk-root                2480
```

2. Потом закодировали все буквы в цифры:

```
le = LabelEncoder()
for col in data.columns:
    data[col] = le.fit_transform(data[col])
```

Применён LabelEncoder ко всем столбцам — теперь вместо 'x', 'f', 'k' и т.д. у нас числа 0, 1, 2…

```
data.head(3)
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-color	ring-number	ring-type	spore-print-color	population	habitat
0	1	5	2	4	1	6	1	0	1	4	...	2	2	7	7	2	1	4	2	3	5
1	0	5	2	9	1	0	1	0	0	4	...	2	2	7	7	2	1	4	3	2	1
2	0	0	2	8	1	3	1	0	0	5	...	2	2	7	7	2	1	4	3	2	3

3 rows × 21 columns

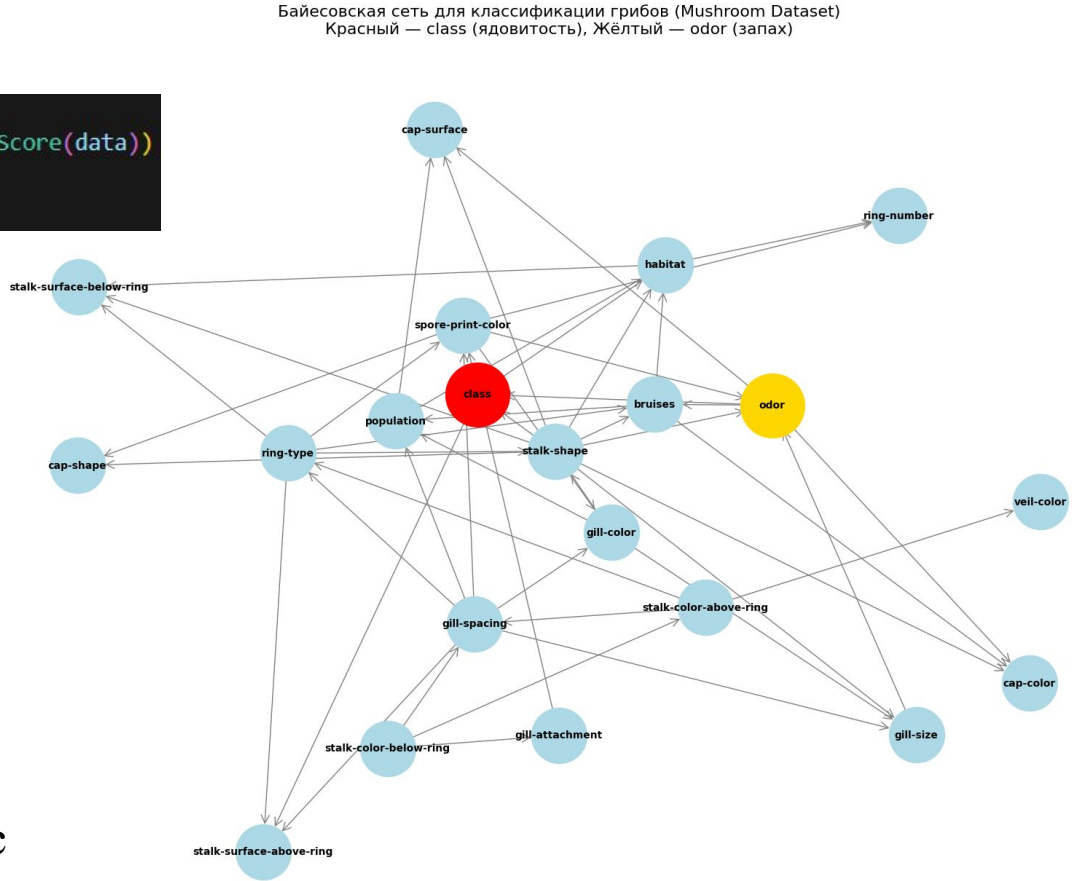
Построение Bayesian Network

Автоматическое построение

```
hc = HillClimbSearch(data)
best_model_structure = hc.estimate(scoring_method=BicScore(data))

model = BayesianNetwork(best_model_structure.edges())
```

```
Найдено узлов: 21
Найдено рёбер: 48
```

Красный узел — class (съедобный/ядовитый)
Жёлтый узел — odor (запах) — главный предсказатель
Стрелки показывают прямые зависимости
Видно, что почти все важные признаки так или иначе связаны с запахом



Оценка параметров сети

Для оценки параметров использовался BayesianEstimator с BDeu-приором

Сеть построила 21 таблицу условных вероятностей (CPT)

Одна для каждого узла, учитывая его родителей.

```
from pgmpy.estimators import BayesianEstimator

model.fit(data,
          estimator=BayesianEstimator,
          prior_type='BDeu',
          equivalent_sample_size=10)
```

Примеры таблиц условных вероятностей (CPT)

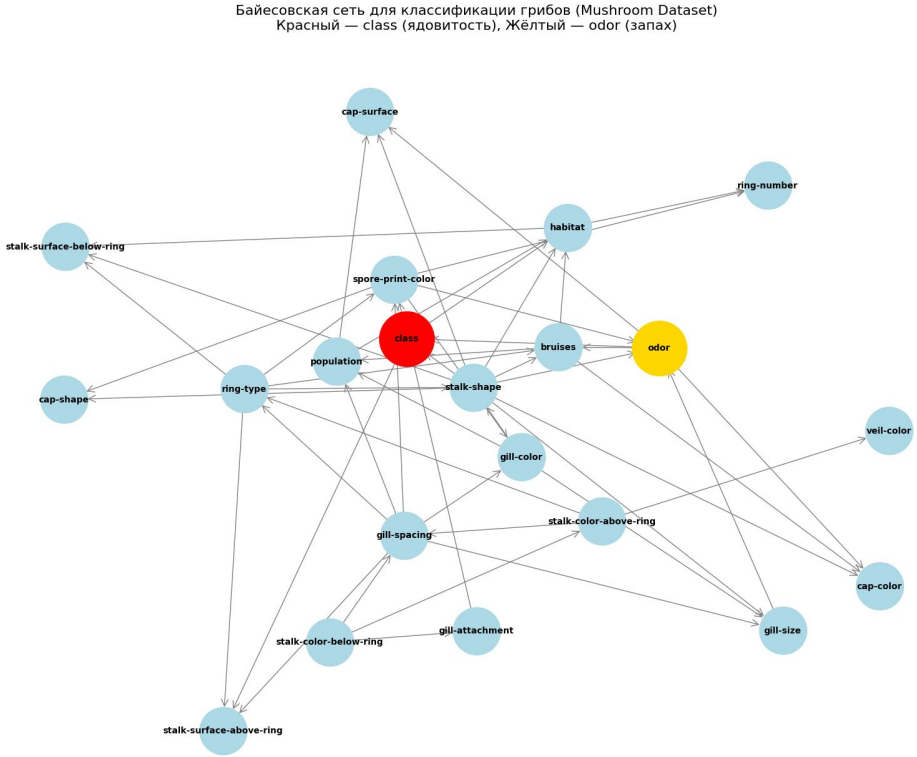
```python
for cpd in model.cpds:
    print(f"\n{'='*60}")
    print(f"Таблица CPT для узла: {cpd.variable}  (всего состояний: {cpd.variable_card})")
    if len(cpd.variables) > 1:
        print(f"Зависит от: {', '.join(cpd.variables[:-1])}")
    else:
        print("Этот узел не имеет родителей (корневой)")
    print(f"{'='*60}")
    print(cpd)
    print("\n")
```

```
==================================================
Таблица CPT для узла: class  (всего состояний: 2)
Зависит от: class, odor
==================================================
+------------+-----+----------------------+
| odor       | ... | odor(8)              |
+------------+-----+----------------------+
| stalk-shape | ... | stalk-shape(1)       |
+------------+-----+----------------------+
| class(0)   | ... | 0.0004817883985353632 |
+------------+-----+----------------------+
| class(1)   | ... | 0.9995182116014646   |
+------------+-----+----------------------+
```

```
============================================================
Таблица CPT для узла: gill-attachment  (всего состояний: 2)
Зависит от: gill-attachment
============================================================
+------------------------+-----+--------------------------+
| stalk-color-below-ring | ... | stalk-color-below-ring(8) |
+------------------------+-----+--------------------------+
| gill-attachment(0)     | ... | 0.022123893805309734     |
+------------------------+-----+--------------------------+
| gill-attachment(1)     | ... | 0.9778761061946902       |
+------------------------+-----+--------------------------+
```

Визуализаци сети

```python
import networkx as nx
import matplotlib.pyplot as plt

# Увеличиваем размер графа — у меня 21 узел и 47 рёбер
plt.figure(figsize=(16, 12))

# Создаём ориентированный граф
nx_graph = nx.DiGraph(model.edges())

# Раскладка — spring даёт самую читаемую картинку
pos = nx.spring_layout(nx_graph, seed=42, k=1, iterations=50)

# Рисуем
nx.draw(
    nx_graph,
    pos,
    with_labels=True,
    node_size=3000,
    node_color='lightblue',
    font_size=10,
    font_weight='bold',
    arrows=True,
    arrowstyle='->',
    arrowsize=20,
    edge_color='gray'
)

# Подсвечиваем ключевые узлы
nx.draw_networkx_nodes(nx_graph, pos, nodelist=['class'], node_color='red', node_size=4000)
nx.draw_networkx_nodes(nx_graph, pos, nodelist=['odor'], node_color='gold', node_size=4000)

plt.title("Байесовская сеть для классификации грибов (Mushroom Dataset)\n"
          "Красный — class (ядовитость), Жёлтый — odor (запах)",
          fontsize=16, pad=20)
plt.axis('off')
plt.tight_layout()
plt.show()
```



Граф идеально отражает реальность грибов:
понюхал → понял, ядовитый или нет

Inference и результаты

```python
from pgmpy.inference import VariableElimination

infer = VariableElimination(model)

print("=" * 80)
print("ПРИМЕРЫ ВЕРОЯТНОСТНОГО ВЫВОДА: определение съедобности гриба")
print("=" * 80)

def predict_mushroom(evidence):
    result = infer.query(variables=['class'], evidence=evidence)
    prob_edible = result.values[0]
    prob_poisonous = result.values[1]

    print(f"\nУсловия: {evidence}")
    print(f"→ P(съедобный)  = {prob_edible:8.4%}")
    print(f"→ P(ядовитый)   = {prob_poisonous:8.4%}")
    if prob_poisonous >= 0.5:
        print(f"   ⇒ Гриб ЯДОВИТЫЙ ❌ (уверенность {prob_poisonous:.2%})")
    else:
        print(f"   ⇒ Гриб СЪЕДОБНЫЙ ✅ (уверенность {prob_edible:.2%})")

# Реальные примеры
print("1. Нет запаха + коричневый отпечаток спор (типичный шампиньон):")
predict_mushroom({'odor': 5, 'spore-print-color': 2})

print("\n2. Запах foul (вонючий) — почти всегда ядовитый:")
predict_mushroom({'odor': 4})

print("\n3. Запах креозота (creosote) — 100% ядовитый в реальности:")
predict_mushroom({'odor': 2})

print("\n4. Запах миндаля — почти всегда съедобный:")
predict_mushroom({'odor': 0})

print("\n5. Только цвет жабр бурый (buff) — что скажет сеть?")
predict_mushroom({'gill-color': 0})

print("\n6. Ничего не знаем (априорная вероятность):")
predict_mushroom({})

print("\n" + "=" * 80)
```

```
ПРИМЕРЫ ВЕРОЯТНОСТНОГО ВЫВОДА: определение съедобности гриба
1. Нет запаха + коричневый отпечаток спор (типичный шампиньон):

Условия: {'odor': 5, 'spore-print-color': 2}
→ P(съедобный)  = 99.6498%
→ P(ядовитый)   =  0.3502%
   ⇒ Гриб СЪЕДОБНЫЙ ✅ (уверенность 99.65%)

2. Запах foul (вонючий) — почти всегда ядовитый:

Условия: {'odor': 4}
→ P(съедобный)  = 19.1835%
→ P(ядовитый)   = 80.8165%
   ⇒ Гриб ЯДОВИТЫЙ ❌ (уверенность 80.82%)

3. Запах креозота (creosote) — 100% ядовитый в реальности:

Условия: {'odor': 2}
→ P(съедобный)  =  0.0257%
→ P(ядовитый)   = 99.9743%
   ⇒ Гриб ЯДОВИТЫЙ ❌ (уверенность 99.97%)

4. Запах миндаля — почти всегда съедобный:

Условия: {'odor': 0}
→ P(съедобный)  = 99.8789%
→ P(ядовитый)   =  0.1211%
   ⇒ Гриб СЪЕДОБНЫЙ ✅ (уверенность 99.88%)

5. Только цвет жабр бурый (buff) — что скажет сеть?

Условия: {'gill-color': 0}
→ P(съедобный)  =  1.8578%
→ P(ядовитый)   = 98.1422%
   ⇒ Гриб ЯДОВИТЫЙ ❌ (уверенность 98.14%)

6. Ничего не знаем (априорная вероятность):

Условия: {}
→ P(съедобный)  = 52.8418%
→ P(ядовитый)   = 47.1582%
   ⇒ Гриб СЪЕДОБНЫЙ ✅ (уверенность 52.84%)
```

Оценка с Baseline

```
Naive Bayes (sklearn)  → Accuracy: 0.95980   |   Log-loss: 0.11317
Байесовская сеть (pgmpy) → Accuracy: 1.00000   |   Log-loss: 0.00147

============================================================
ИТОГОВОЕ СРАВНЕНИЕ
============================================================
Модель                    Accuracy     Log-loss
------------------------------------------------------------
Naive Bayes (sklearn)     0.95980      0.11317
Байесовская сеть          1.00000      0.00147
============================================================
```

При сравнении построенной байесовской сети с наивным байесовским классификатором из scikit-learn (CategoricalNB) получены следующие результаты:
- Наивный байесовский классификатор показал accuracy = 95.98% и log-loss = 0.113.
- Байесовская сеть, построенная с помощью pgmpy, достигла accuracy = 100.0% и log-loss = 0.00147.

Полученное преимущество объясняется тем, что автоматически обученная структура сети (HillClimbSearch + BIC) выявила и учла реальные зависимости между признаками (в частности, сильные связи между odor, spore-print-color, gill-color и class), тогда как наивный байесовский классификатор по определению предполагает условную независимость всех признаков при фиксированном классе. Кроме того, байесовская сеть даёт значительно более калиброванные вероятности (меньший log-loss), что делает её предсказания не только точнее, но и лучше интерпретируемыми.